Scale-Aware Navigation of a Low-Cost Quadrocopter with a Monocular Camera

Jakob Engel, Jürgen Sturm, Daniel Cremers

Department of Computer Science, Technical University of Munich, Germany
{engelj,sturmju,cremers}@in.tum.de

Abstract

We present a complete solution for the visual navigation of a small-scale, low-cost quadrocopter in unknown environments. Our approach relies solely on a monocular camera as the main sensor, and therefore does not need external tracking aids such as GPS or visual markers. Costly computations are carried out on an external laptop that communicates over wireless LAN with the quadrocopter. Our approach consists of three components: a monocular SLAM system, an extended Kalman filter for data fusion, and a PID controller. In this paper, we (1) propose a simple, yet effective method to compensate for large delays in the control loop using an accurate model of the quadrocopter's flight dynamics, and (2) present a novel, closed-form method to estimate the scale of a monocular SLAM system from additional metric sensors. We extensively evaluated our system in terms of pose estimation accuracy, flight accuracy, and flight agility using an external motion capture system. Furthermore, we compared the convergence and accuracy of our scale estimation method for an ultrasound altimeter and an air pressure sensor with filtering-based approaches. The complete system is available as open-source in ROS. This software can be used directly with a low-cost, off-the-shelf Parrot AR.Drone quadrocopter, and hence serves as an ideal basis for follow-up research projects.

Keywords: quadrocopter, visual navigation, visual SLAM, monocular SLAM, scale estimation, AR.Drone

1. Introduction

Research interest in autonomous micro-aerial vehicles (MAVs) has grown rapidly in the past years. Significant progress has been made, and recent examples include aggressive flight manoeuvres [1], collaborative construction tasks [2], ball throwing and catching [3] or the coordination of large fleets of quadrocopters [4]. However, all of these systems require external motion capture systems. Flying in unknown, GPS-denied environments is still an open research problem. The key challenges here are to localize the robot purely from its own sensor data and to robustly navigate it even under temporary sensor outage. This requires both a solution to the so-called simultaneous localization and mapping (SLAM) problem as well as robust state estimation and control methods. These challenges are even more expressed on low-cost hardware with inaccurate actuators, noisy sensors, significant delays, and limited on-board computation resources.

For solving the SLAM problem on MAVs, different types of sensors such as laser range scanners [5], monocular cameras [6], stereo cameras [7], and RGB-D sensors [8, 9] have been explored in the past. In our point of view, monocular cameras have two major advantages over other modalities: (1) they provide rich information at a low weight, power consumption, size, and cost and (2) in contrast to depth sensors, a monocular SLAM system is not intrinsically limited in its visual range, and therefore can operate both in small, confined and large, open spaces. The drawback however is, that the scale of the environment cannot be determined from monocular vision alone, such that additional sensors, such as an IMU or air pressure sensor, are required.

Figure 1: A low-cost quadrocopter navigates in unstructured environments using the front camera as its main sensor. The quadrocopter is able to hold a position, fly figures with absolute scale, and recover from temporary tracking loss. Picture taken at the TUM open day.

In this paper we follow up on our previous work in

[10, 11] where we presented our approach to use a monocular camera to navigate a low-cost quadrocopter in an unknown, unstructured environment. Computations are performed off-board on a ground-based laptop. For our experiments we used both the low-cost Parrot AR.Drone 1.0 and 2.0 which are available for $300 and, with a weight of only 420 g and a protective hull, safe to be used in public places (as illustrated in Fig. 1).

We extend upon our previous work in several ways: First, we provide a more in-depth explanation of the proposed scale estimation method, and show that it can accurately estimate the scale of a monocular SLAM system even from very noisy sensor data such as an air pressure sensor. We provide an extensive evaluation both on synthetic and on real-world data sequences, and perform a direct comparison with current state-of-the-art filtering based methods. Second, we provide additional experimental results, in particular we evaluate the flight and pose estimation accuracy using an external motion capture system as ground truth. Third, we provide the complete system as an open-source ROS package. Our software can be used directly with a low-cost, off-the-shelf Parrot AR.Drone 1.0 or 2.0, and no modifications to hardware or onboard software are required. It therefore serves as an ideal base for follow-up projects.

A video demonstrating the practical performance of our system, its ability to accurately fly to a given position and its robustness to loss of visual tracking is available online:

http://youtu.be/eznMokFQmpc

Further, we provide an open-source ROS implementation of the complete system:

http://ros.org/wiki/tum_ardrone

2. Related Work

Previous work on autonomous flight with quadrocopters can be categorized into two main research areas. Several results have been published where the focus is on accurate and agile quadrocopter control [1, 12]. These works however rely on advanced external tracking systems, restricting their use to a lab environment. A similar approach is to distribute artificial markers in the environment, simplifying pose estimation [13]. Other approaches learn a map offline from a previously recorded, manual flight and thereby enable a quadrocopter to reproduce the same trajectory [14]. For outdoor flights where accurate GPS-based pose estimation is possible, complete solutions are available as commercial products [15].

We are interested in autonomous flight without previous knowledge about the environment or GPS signals, while using only on-board sensors. Previous work on autonomous quadrocopter flight has explored lightweight laser scanners [5], RGB-D sensors [8, 9] or stereo rigs [16] mounted on a quadrocopter as primary sensors. While these sensors provide absolute scale of the environment, their drawback is a limited range and large weight, size, and power consumption when compared to a monocular set-up.

In our work we therefore focus on a monocular camera for pose estimation. Stabilizing controllers based on optical flow from a monocular camera were presented e.g. in [17, 18], and similar methods are integrated in commercially available hardware [19]. These systems however make strong assumptions about the environment such as a flat, horizontal ground plane. Additionally, they are subject to drift over time, and are therefore not suited for long-term autonomous navigation.

To eliminate drift, various monocular SLAM methods have been investigated on quadrocopters, both with off-board [5] and on-board processing [6, 20, 21]. A particular challenge for monocular SLAM is that the scale of the map needs to be estimated from additional metric sensors such as an air pressure sensor as it cannot be recovered from vision alone. This problem has been addressed in recent publications such as [22], where the scale is added to the extended Kalman filter as an additional state variable. In contrast to this, we propose in this paper a novel approach which directly computes the unknown scale factor from a set of observations: using a statistical formulation, we derive a closed-form, consistent estimator for the scale of the visual map. Our method yields accurate and robust results both in simulation and practice. As metric sensors we evaluated both an air pressure sensor as well as an ultrasound altimeter. The proposed method can be used with any monocular SLAM algorithm and sensors providing metric position or velocity measurements.

In contrast to previous work [6], we deliberately refrain from using expensive, customized hardware: the only hardware required is the AR.Drone, which comes at a costs of merely $300 – a fraction of the cost of quadrocopters used in previous work. Released in 2010 and marketed as a high-tech toy, it has been used and discussed in several research projects [23, 24, 25].

The remainder of this article is organized as follows: in Sec. 3, we briefly introduce the Parrot AR.Drone and its sensors. In Sec. 4 we derive the proposed maximum-likelihood estimator for the scale of a monocular SLAM system. We also describe in detail all necessary prepro-

cessing steps as well as how the variances can be estimated from the data. In Sec. 5 we describe our approach as a whole, in particular we describe the EKF, how we estimate the required model parameters and how we compensate for time delays. In Sec. 6 we present an extensive evaluation of our scale estimation method and compare it to a state-of-the-art filtering based method. We also provide extensive experimental results on the flight agility, accuracy and robustness of our system using an external motion capture system. Finally, we conclude the paper in Sec. 7 with a summary and outlook to future work.

3. Hardware Platform

For the experiments we use the Parrot AR.Drone 2.0, a commercially available quadrocopter as platform. Compared to other modern MAVs such as Ascending Technology's Pelican or Hummingbird quadrocopters, its main advantages are the low price, its robustness to crashes, and the fact that it can safely be used indoor and close to people. This however comes at the price of flexibility: Neither the hardware itself nor the software running onboard can easily be modified, and communication with the quadrocopter is only possible over wireless LAN. With battery and hull, the AR.Drone measures 53 cm × 52 cm and weights 420 g.

3.1. Sensors

The AR.Drone 2.0 is equipped with a 3-axis gyroscope and accelerometer, an ultrasound altimeter and two cameras. Furthermore it features an air pressure sensor and a magnetic compass. The first camera is aimed forward, covers a diagonal field of view of 92°, has a resolution of 640×360, significant radial distortion and a rolling shutter. The captured video is streamed to a laptop at 30 fps, using lossy compression. The second camera aims downward, covers a diagonal field of view of 64° and has a resolution of 320×240 at 60 fps. Only one of the two video streams can be streamed to to the laptop at the same time. The on-board software uses the down-looking camera to estimate the horizontal velocity, the accuracy of these velocity estimates highly depends on the ground texture and flight altitude. All sensor measurements as well as the estimated horizontal velocities are sent to the laptop at a frequency of up to 200 Hz.

3.2. Control

The on-board software uses the available sensors to control the roll Φ and pitch Θ, the yaw rotational speed $\dot{\Psi}$ and the vertical velocity $\dot{z}$ of the quadrocopter according to an external reference value. This reference is set by sending control commands $\mathbf{u} = (\bar{\Phi}, \bar{\Theta}, \bar{\dot{z}}, \bar{\dot{\Psi}}) \in [-1,1]^4$ to the quadrocopter at a frequency of 100 Hz.

4. Scale Estimation for Monocular SLAM

In this paper, we propose a closed-form solution to estimate the scale $\lambda \in \mathbb{R}^+$ of a monocular SLAM system. For this, we assume that the robot makes noisy measurements of absolute distances or velocities from one or more metric sensors such as an ultrasound altimeter or an air pressure sensor. In this chapter we first formulate the problem statistically in 4.1, and then discuss several straightforward estimation strategies and demonstrate why they lead to a biased result in 4.2. In 4.3 we derive the maximum likelihood estimator for the scale λ. In 4.4 we describe the required data preprocessing steps and in 4.5 how the required parameters are estimated from the data.

4.1. Problem Formulation

As a first step, the quadrocopter measures in regular intervals the d-dimensional distance travelled within a certain timespan, according to the visual SLAM system $\mathbf{x}_i \in \mathbb{R}^d$ and using the metric sensors available $\mathbf{y}_i \in \mathbb{R}^d$. Note that for an altimeter $d = 1$, as only vertical movement can be measured – other sensor types however can measure the full 3D translation (such as GPS). Each interval gives a sample pair $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{x}_i$ is scaled according to the visual map and $\mathbf{y}_i$ is in metric units. As both $\mathbf{x}_i$ and $\mathbf{y}_i$ measure the motion of the quadrocopter, they are related according to $\mathbf{x}_i \approx \lambda \mathbf{y}_i$.

More specifically, if we assume isotropic, Gaussian white noise in the sensor measurements[1], we obtain a set of sample pairs $\{(\mathbf{x}_1, \mathbf{y}_1) \dots (\mathbf{x}_n, \mathbf{y}_n)\}$ with

$$\begin{aligned} \mathbf{x}_i &\sim \mathcal{N}(\lambda \boldsymbol{\mu}_i, \sigma_x^2 \mathbf{I}_{d\times d}) \\ \mathbf{y}_i &\sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma_y^2 \mathbf{I}_{d\times d}) \end{aligned} \tag{1}$$

where $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_i \in \mathbb{R}^d$ denote the true (unknown) distances and $\sigma_x^2, \sigma_y^2 \in \mathbb{R}^+$ the variances of the measurement errors. Note that the true distances $\boldsymbol{\mu}_i$ are not constant but correspond to the actual distance travelled by the quadrocopter in the respective measurement interval. This preprocessing step is further explained in 4.4.

[1] The noise in $\mathbf{x}_i$ does not depend on λ as it is proportional to the average keypoint depth, which we normalize to be 1 in the first keyframe.

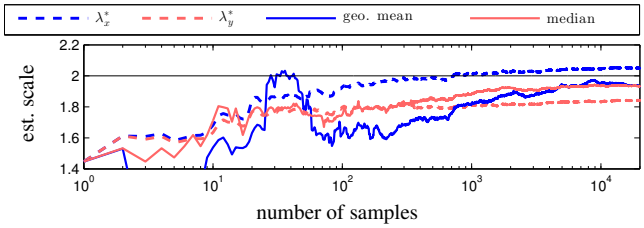


Figure 2: Naïve Estimators: The plot shows the estimated scale as more samples are added for synthetic data. Observe how all Estimators converge to a wrong value, even after adding 20.000 samples (note the logarithmic y axis). For this plot we used $\lambda = 2$, $\sigma_x = 0.3$, $\sigma_y = 0.3$ and $\boldsymbol{\mu}_i \sim \mathcal{N}(0,1)$.

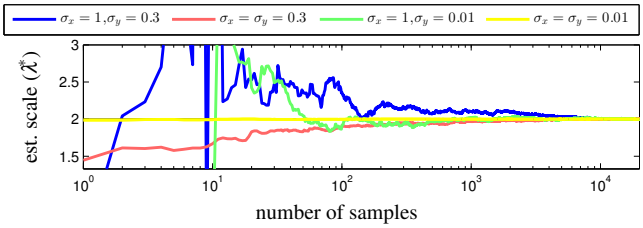


Figure 3: Proposed ML Estimator: The plot shows the estimated scale as more samples are added for synthetic data, using different noise levels. The red line corresponds to the same parameter settings as in Fig. 2. Again, we choose $\lambda = 2$ and $\boldsymbol{\mu}_i \sim \mathcal{N}(0,1)$.

4.2. Naïve Estimation Strategies

A direct, naïve approach to estimating λ from such a set of samples is to compute the arithmetic average, geometric average, or median of the set of quotients $\frac{\|\mathbf{x}_i\|}{\|\mathbf{y}_i\|}$. This approach however fails, as illustrated by the following example: Imagine $\lambda = 1$, no measurement noise on the $\mathbf{x}_i$ and only two sample pairs: $(\mathbf{x}_1, \mathbf{y}_1) = (1, 0.5)$ and $(\mathbf{x}_2, \mathbf{y}_2) = (1, 1.5)$. Although $\mathbf{y}_1$ and $\mathbf{y}_2$ deviate symmetrically from the true value, neither the geometric nor the arithmetic mean of the two quotients estimates the scale correctly: $\frac{1}{2}(\frac{1}{1.5} + \frac{1}{0.5}) \approx 1.3$ and $(\frac{1}{1.5} \cdot \frac{1}{0.5})^{0.5} \approx 1.15$.

Another approach is to minimize the sum of squared differences (SSD) between the re-scaled measurements, i.e., to compute one of the following:

$$\lambda_y^* := \arg\min_{\lambda} \sum_{i=1}^{n} \|\mathbf{x}_i - \lambda \mathbf{y}_i\|^2 = \frac{\sum_i \mathbf{x}_i^T \mathbf{y}_i}{\sum_i \mathbf{y}_i^T \mathbf{y}_i} \tag{2}$$

$$\lambda_x^* := \left(\arg\min_{\lambda} \sum_{i=1}^{n} \|\lambda \mathbf{x}_i - \mathbf{y}_i\|^2 \right)^{-1} = \frac{\sum_i \mathbf{x}_i^T \mathbf{x}_i}{\sum_i \mathbf{x}_i^T \mathbf{y}_i}. \tag{3}$$

The difference between these two lines is whether one aims at scaling the $\mathbf{x}_i$ to the $\mathbf{y}_i$ or vice versa. Both approaches however suffer from the same problem, that is they do not converge to the true scale λ when adding more samples. To study this effect in more detail, we applied these naïve approaches to artificially generated data according to (1) and a wide range of parameter settings. An example result is shown in Fig. 2: All of the above estimation strategies are clearly inconsistent, i.e. they do not converge to the correct value for $n \to \infty$.

4.3. Maximum Likelihood Solution

Maximum-likelihood estimation is a widely used method to estimate unknown parameters of a statistical model. The core idea is to choose the unknown parameter such that the probability of observing the data is maximised. Typically, one minimizes the negative log-likelihood, i.e.,

$$\mathcal{L}(\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_n, \lambda) \propto \frac{1}{2} \sum_{i=1}^{n} \left(\frac{\|\mathbf{x}_i - \lambda \boldsymbol{\mu}_i\|^2}{\sigma_x^2} + \frac{\|\mathbf{y}_i - \boldsymbol{\mu}_i\|^2}{\sigma_y^2} \right) \tag{4}$$

By first minimizing over $\boldsymbol{\mu}_1 \dots \boldsymbol{\mu}_n$ and then over λ, it can be shown analytically that (4) has a unique, global minimum at

$$\boldsymbol{\mu}_i^* = \frac{\lambda^* \sigma_y^2 \mathbf{x}_i + \sigma_x^2 \mathbf{y}_i}{\lambda^{*2} \sigma_y^2 + \sigma_x^2} \tag{5}$$

$$\lambda^* = \frac{s_{xx} - s_{yy} + \text{sign}(s_{xy}) \sqrt{(s_{xx} - s_{yy})^2 + 4 s_{xy}^2}}{2 \sigma_x^{-1} \sigma_y s_{xy}} \tag{6}$$

with $s_{xx} := \sigma_y^2 \sum_{i=1}^n \mathbf{x}_i^T \mathbf{x}_i$, $s_{yy} := \sigma_x^2 \sum_{i=1}^n \mathbf{y}_i^T \mathbf{y}_i$, and $s_{xy} := \sigma_y \sigma_x \sum_{i=1}^n \mathbf{x}_i^T \mathbf{y}_i$ [2]. Together, these equations give a closed-form solution for the ML estimator of λ, assuming the variances σ_x^2 and σ_y^2 are known. It can easily be shown that this solution has the following two important properties:

1. λ^* always lies in between λ_x^* and λ_y^*, and
2. $\lambda^* \to \lambda_x^*$ for $\sigma_x^2 \to 0$, and $\lambda^* \to \lambda_y^*$ for $\sigma_y^2 \to 0$, i.e., these naïve estimators correspond to the extreme case when one of the measurement sources is noise-free.

This leads to the observation that λ^* correctly interpolates between the two extreme cases when one measurement source is noise-free, based on the variances of the two measurement sources. For the full minimization and a derivation of these two properties we refer to [27]. Again

[2] assuming $s_{xy} > 0$, which holds for a sufficiently large sample set and $\lambda > 0$

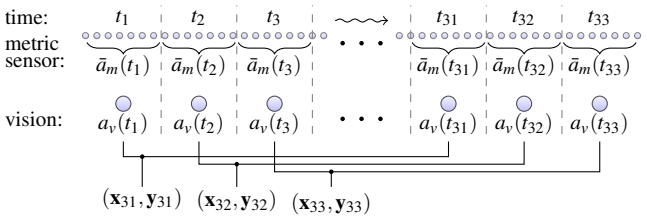


Figure 4: Computing the Set of Sample Pairs: For each visual pose estimate, we generate one sample pair $(\mathbf{x}_i, \mathbf{y}_i)$, consisting of the visual and the metric vertical distance travelled within the last second.

we applied this estimator to artificially generated data according to (1). Figure 3 shows the result for different parameter settings. As can be seen in the figure, the proposed method always converges to the correct value, even for high noise levels. A more extensive evaluation of the accuracy of the proposed estimator, as well as a direct comparison with a filtering-based approach on both synthetic and real-world data using different sensor modalities will be presented in Sec. 6.1.

4.4. Generating the sample set

We use the derived method to estimate the scale of a visual SLAM system (operating at 30 Hz) using a metric sensor such as an air pressure sensor (operating at 200 Hz). In order to compute the set of sample pairs $\{(\mathbf{x}_1, \mathbf{y}_1) \dots (\mathbf{x}_n, \mathbf{y}_n)\}$, we do the following:

1. For each visual altitude measurement $a_v(t_i) \in \mathbb{R}$, we compute a corresponding metric altitude measurement $\bar{a}_m(t_i) \in \mathbb{R}$ by averaging over a small window of raw sensor measurements. The window size is chosen such that each raw measurement is used at most once, to preserve statistical independence while reducing measurement noise.
2. For each $a_v(t_i)$, we compute the visual distance travelled within a certain timespan of k frames $\mathbf{x}_i := a_v(t_i) - a_v(t_{i-k})$, as well as the metric distance travelled within that timespan $\mathbf{y}_i := \bar{a}_m(t_i) - \bar{a}_m(t_{i-k})$.

Figure 4 illustrates this process. The result is one statistically independent sample pair for each visual pose estimate, i.e. 30 samples per second. The window size k can be chosen freely: if the quadrocopter moves fast this value can be small, if it moves slower it should be increased. In practice we found that a value of $k \in [30, 60]$, corresponding to one to two seconds, gives good results. To increase robustness, k can also be chosen differently for each frame, depending on the current speed of the quadrocopter.

4.5. Estimation of the Measurement Variances

The computation of λ^* requires the variances of the metric sensor σ_m^2 and the vision estimates σ_v^2. These variances can be estimated from the data as follows: Under the assumptions that (1) consecutive measurements are independent and identically distributed, (2) measurements are taken at regular time intervals, and (3) the quadrocopter's velocity is approximately constant over three consecutive measurements, we obtain

$$a_v(t_{i-1}) \sim \mathcal{N}(a-b, \sigma_v^2) \tag{7}$$
$$a_v(t_i) \sim \mathcal{N}(a, \sigma_v^2) \tag{8}$$
$$a_v(t_{i+1}) \sim \mathcal{N}(a+b, \sigma_v^2) \tag{9}$$

where $a_v(t_i)$ is the observed altitude at time-step t_i, and a and b are the (unknown) true height and velocity respectively. From the additivity of the normal distribution it follows that

$$(a_v(t_{i-1}) - 2a_v(t_i) + a_v(t_{i+1})) \sim \mathcal{N}(0, \sigma_v^2 + 4\sigma_v^2 + \sigma_v^2). \tag{10}$$

Hence, σ_v^2 can be estimated from a series of height measurements $a_v(t_1), \dots, a_v(t_n)$ using

$$\sigma_v^{2*} = \frac{1}{6}\frac{1}{n-3}\sum_{i=2}^{n-1}(a_v(t_{i-1}) - 2a_v(t_i) + a_v(t_{i+1}))^2 \tag{11}$$

The variance of $\mathbf{x}_i$ is then given by $\sigma_x^2 = 2\sigma_v^2$. The same method is used to estimate σ_y^2.

Note that we assume that the pose estimation variance is constant: while for an air pressure sensor this is a reasonable assumption, its validity in case of a visual SLAM system is questionable, as the pose error depends on many factors which might change throughout the flight – in particular on the number and depth of the observed keypoints. In practice however, the accuracy of the visual SLAM system varies only slightly, and the measurement error of an altitude pressure sensor dominates by several orders of magnitude.

5. Visual Navigation System

In this chapter we first give an overview over the developed system and its core components in 5.1. We then detail the developed EKF, its observation and prediction model and how we compensate for delays arising from off-board computation in 5.2 and 5.3. We then show how model parameters, which depend on the actual quadrocopter used, can be estimated from test-flight data in 5.5.

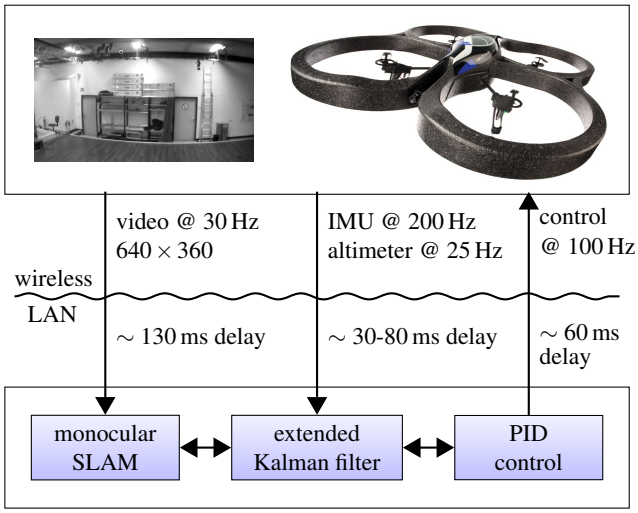


Figure 5: Approach Outline: Our navigation system consists of three major components: a monocular SLAM implementation for visual tracking, an EKF for data fusion and prediction, and PID control for pose stabilization and navigation. All computations are performed off-board, which leads to significant, varying delays which our approach has to compensate.

5.1. System Overview

The system consists of three main components as shown in Fig. 5:

1) Monocular SLAM: Our solution is based on Parallel Tracking and Mapping (PTAM) [26]. After map initialization, we rotate the visual map such that the xy-plane corresponds to the horizontal plane according to the accelerometer data, and scale it such that the average keypoint depth is 1. Throughout tracking, the scale of the map $\lambda \in \mathbb{R}$ is estimated as described in Sec. 4. Furthermore, we use the pose estimates from the EKF to identify and reject falsely tracked frames as well as to assist relocalization after tracking loss.

2) Extended Kalman Filter: In order to fuse all available data, we employ an EKF, which includes a full motion model of the quadrocopter's flight dynamics and reaction to control commands. The EKF is also used to compensate for time delays in the system.

3) PID Control: Based on the position and velocity predictions from the EKF, we apply PID control to steer the quadrocopter towards the desired goal location $\mathbf{p} = (\hat{x},\hat{y},\hat{z},\hat{\Psi})^T \in \mathbb{R}^4$ in a global coordinate system. According to the current state estimate, we transform the generated controls into a robot-centric coordinate frame and send them to the quadrocopter. For each of the four degrees of freedom, we employ a separate PID controller for which we experimentally determined suitable controller gains.

5.2. EKF Prediction and Observation

The state space consists of a total of ten state variables

$$\mathbf{x}_t := (x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \Phi_t, \Theta_t, \Psi_t, \dot{\Psi}_t)^T \in \mathbb{R}^{10}, \quad (12)$$

where (x_t, y_t, z_t) denotes the position of the quadrocopter in meter and $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ the velocity in meter per second, both in world coordinates. Further, the state contains the roll Φ_t, pitch Θ_t and yaw Ψ_t angle of the quadrocopter in degree, as well as the yaw-rotational speed $\dot{\Psi}_t$ in degree per second. In the following, we define for each sensor an observation function $h(\mathbf{x}_t)$ and describe how the respective observation vector $\mathbf{z}_t$ is derived from the sensor readings.

5.2.1. Odometry Observation Model

The quadrocopter measures its horizontal speed $\hat{v}_{x,t}$ and $\hat{v}_{y,t}$ in its local coordinate frame, which we transform into the global frame $\dot{x}_t$ and $\dot{y}_t$. The roll and pitch angles $\hat{\Phi}_t$ and $\hat{\Theta}_t$ measured by the accelerometer are direct observations of Φ_t and Θ_t. To account for yaw-drift and uneven ground, we differentiate the height measurements $\hat{h}_t$ and yaw measurements $\hat{\Psi}_t$ and treat them as observations of the respective velocities. The resulting observation function $h_{\mathrm{I}}(\mathbf{x}_t)$ and measurement vector $\mathbf{z}_{\mathrm{I},t}$ is hence given by

$$h_{\mathrm{I}}(\mathbf{x}_t) := \begin{pmatrix} \dot{x}_t \cos\Psi_t - \dot{y}_t \sin\Psi_t \\ \dot{x}_t \sin\Psi_t + \dot{y}_t \cos\Psi_t \\ \dot{z}_t \\ \Phi_t \\ \Theta_t \\ \dot{\Psi}_t \end{pmatrix} \quad (13)$$

$$\mathbf{z}_{\mathrm{I},t} := (\hat{v}_{x,t}, \hat{v}_{y,t}, \frac{\hat{h}_t - \hat{h}_{t-1}}{\delta_{t-1}}, \hat{\Phi}_t, \hat{\Theta}_t, \frac{\hat{\Psi}_t - \hat{\Psi}_{t-1}}{\delta_{t-1}})^T \quad (14)$$

where δ_t denotes the time passed from timestep t to $t+1$.

5.2.2. Visual Observation Model

When PTAM successfully tracks a video frame, we first scale the pose estimate with the estimated scaling factor λ^* and transform it to the coordinate system of the quadrocopter, leading to a direct observation of the quadrocopter's pose given by

$$h_{\mathrm{P}}(\mathbf{x}_t) := (x_t, y_t, z_t, \Phi_t, \Theta_t, \Psi_t)^T \quad (15)$$

$$\mathbf{z}_{\mathrm{P},t} := f(\mathbf{E}_{\mathcal{DC}} \mathbf{E}_{\mathcal{C},t}) \quad (16)$$

where $\mathbf{E}_{\mathcal{C},t} \in \mathrm{SE}(3)$ is the estimated camera pose (scaled with λ), $\mathbf{E}_{\mathcal{DC}} \in \mathrm{SE}(3)$ the constant transformation from the camera to the quadrocopter coordinate system, and $f : \mathrm{SE}(3) \to \mathbb{R}^6$ the conversion from SE(3) to the roll-pitch-yaw representation.

5.2.3. Prediction Model

The prediction model describes how the state vector $\mathbf{x}_t$ evolves from one time step to the next. In particular, we approximate the quadrocopter's horizontal acceleration $\ddot{x}, \ddot{y}$ based on its current state $\mathbf{x}_t$, and estimate its vertical acceleration $\ddot{z}$, yaw-rotational acceleration $\ddot{\Psi}$ and roll/pitch rotational speed $\dot{\Phi}, \dot{\Theta}$ based on the state $\mathbf{x}_t$ and the active control $\mathbf{u}_t$.

The horizontal acceleration is proportional to the horizontal force acting upon the quadrocopter, given by

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} \propto \mathbf{f}_{\text{acc}} - \mathbf{f}_{\text{drag}} \tag{17}$$

where $\mathbf{f}_{\text{drag}}$ denotes the drag and $\mathbf{f}_{\text{acc}}$ denotes the accelerating force. In general, the drag force has a linear and a quadratic component, corresponding to laminar and turbulent flow – given the comparatively low movement speed of the quadrocopter however, we can safely approximate it by a purely linear function of the current horizontal velocity. The accelerating force $\mathbf{f}_{\text{acc}}$ is proportional to the projection of the quadrocopter's z-axis onto the horizontal plane. This leads to

$$\ddot{x}(\mathbf{x}_t) = c_1 R(\Phi_t, \Theta_t, \Psi_t)_{1,3} - c_2 \dot{x}_t \tag{18}$$

$$\ddot{x}(\mathbf{y}_t) = c_1 R(\Phi_t, \Theta_t, \Psi_t)_{2,3} - c_2 \dot{x}_t \tag{19}$$

where $R(\cdot)_{i,j}$ denotes the entries in the rotation matrix defined by the roll, pitch and yaw angles.

Note that this model assumes that the overall thrust generated by the four rotors is constant. Furthermore, we approximate the influence of sent controls $\mathbf{u}_t = (\bar{\Phi}_t, \bar{\Theta}_t, \bar{\dot{z}}_t, \bar{\dot{\Psi}}_t)$ with a linear model:

$$\dot{\Phi}(\mathbf{x}_t, \mathbf{u}_t) = c_3 \bar{\Phi}_t - c_4 \Phi_t \tag{20}$$

$$\dot{\Theta}(\mathbf{x}_t, \mathbf{u}_t) = c_3 \bar{\Theta}_t - c_4 \Theta_t \tag{21}$$

$$\ddot{\Psi}(\mathbf{x}_t, \mathbf{u}_t) = c_5 \bar{\dot{\Psi}}_t - c_6 \dot{\Psi}_t \tag{22}$$

$$\ddot{z}(\mathbf{x}_t, \mathbf{u}_t) = c_7 \bar{\dot{z}}_t - c_8 \dot{z}_t \tag{23}$$

The overall state transition function is now given by

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \\ \dot{x}_{t+1} \\ \dot{y}_{t+1} \\ \dot{z}_{t+1} \\ \Phi_{t+1} \\ \Theta_{t+1} \\ \Psi_{t+1} \\ \dot{\Psi}_{t+1} \end{pmatrix} \leftarrow \begin{pmatrix} x_t \\ y_t \\ z_t \\ \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \Phi_t \\ \Theta_t \\ \Psi_t \\ \dot{\Psi}_t \end{pmatrix} + \delta_t \begin{pmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \\ \ddot{x}(\mathbf{x}_t) \\ \ddot{y}(\mathbf{x}_t) \\ \ddot{z}(\mathbf{x}_t, \mathbf{u}_t) \\ \dot{\Phi}(\mathbf{x}_t, \mathbf{u}_t) \\ \dot{\Theta}(\mathbf{x}_t, \mathbf{u}_t) \\ \dot{\Psi}_t \\ \ddot{\Psi}(\mathbf{x}_t, \mathbf{u}_t) \end{pmatrix} \tag{24}$$

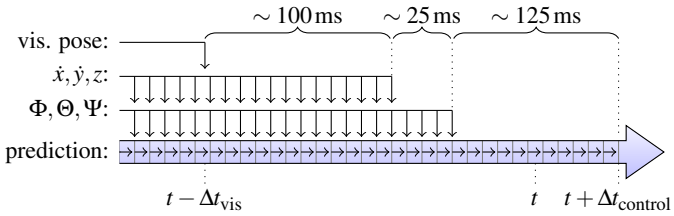


Figure 6: Pose Prediction: Measurements and control commands arrive with significant delays. To compensate for these delays, we keep a history of observations and sent control commands between $t - \Delta t_{\text{vis}}$ and $t + \Delta t_{\text{control}}$ and re-calculate the EKF state when required. Note the large timespan with no or only partial sensor observations.

using the model specified in (18) to (23). Note that, due to the many assumptions made, we do not claim the physical correctness of this model. However, as we will show in Sec. 6, it performs very well in practice: the behaviour of all state parameters and the effect of all control commands is approximated, allowing "blind" prediction, i.e., prediction without observations, for a brief period of time. This is an important prerequisite for effective delay compensation as described in the following section.

5.3. Delay Compensation

For controlling a quickly reacting system such as a quadrocopter, not only an accurate but also a delay-free state estimate is necessary, as delays quickly provoke oscillations and unstable behaviour. In the considered system we observe time delays of up to 250 ms between the moment a video frame is captured and the moment a control command based on this video frame reaches the quadrocopter. Furthermore, different sensor modalities have different delays which need to be synchronized properly.

We found that height and horizontal velocity measurements arrive with the same delay, which is slightly larger than the delay of attitude measurements. The delay of visual pose estimates Δt_{vis} is by far the largest. Furthermore we account for the time required for a new control command to reach the quadrocopter $\Delta t_{\text{control}}$. All timing values given subsequently are typical values, the exact values depend on the wireless connection quality and are determined by a combination of regular ICMP echo requests sent to the quadrocopter and offline calibration experiments as further detailed in [27]. As we deliberately refrain from modifying the on-board hardware or software – which currently does not provide timing information – we only use timestamps from the ground-station.

Our approach works as follows: first, we time-stamp all incoming data and store it in an observation buffer. Control commands are then calculated using a prediction for

the quadrocopter's pose at $t + \Delta t_{\text{control}}$, where t is the current time. For this prediction, we start with the saved state of the EKF at $t - \Delta t_{\text{vis}}$ (i.e., after the last visual observation/unsuccessfully tracked frame). Subsequently, we predict ahead up to $t + \Delta t_{\text{control}}$, integrating previously issued control commands and stored sensor measurements as observations. Figure 6 illustrates this process. Note the large timespan of up to 125 ms that needs to be bridged without observations available: during this timespan the quadrocopter's state is propagated solely based on the described prediction model, incorporating previously issued control commands. In the long run on the other hand, the EKF state is dominated by the pose estimates from the visual SLAM system due to their superior accuracy.

With this approach, we are able synchronize different sensor modalities and to compensate for delayed and missing observations without loosing information, at the expense of recalculating the last cycles of the EKF. As the state only has ten dimensions, many of which are independent, the computational cost of this is negligible compared to the visual tracking.

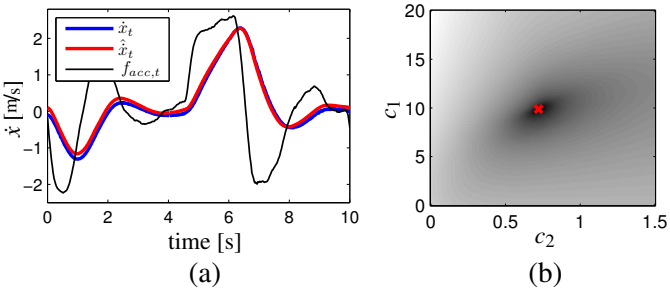
5.4. Control

We employ a simple PID controller to generate the control signals, which are then sent to the quadrocopter at 100 Hz. Each control signal defines the desired roll $\bar{\Phi}$ and pitch $\bar{\Theta}$ angle, the yaw rotational speed $\bar{\dot{\Psi}}$ and the vertical velocity $\bar{\dot{z}}$; each as a fraction of the respective maximal allowed value[3]. This corresponds exactly to the information provided by a human pilot remote-controlling the quadrocopter via smartphone.

Given a target position $\mathbf{p} = (\hat{x}, \hat{y}, \hat{z}, \hat{\Psi})^T$ and the predicted quadrocopter state $\mathbf{x}_{t+\Delta t_{\text{control}}}$, we apply separate PID control to all four controllable degrees of freedom, rotating the result to match the quadrocopter's yaw orientation. For readability, we omit the time indices and already include our experimentally found optimal control gains:

$$\begin{pmatrix} \bar{\Phi} \\ \bar{\Theta} \\ \bar{\dot{z}} \\ \bar{\dot{\Psi}} \end{pmatrix} = \begin{pmatrix} R(\Psi) \begin{bmatrix} 0.5\,(\hat{x}-x) + 0.32\,\dot{x} + 0 \\ 0.5\,(\hat{y}-y) + 0.32\,\dot{y} + 0 \end{bmatrix} \\ 0.6\,(\hat{z}-z) + 0.2\,\dot{z} + 0.01\int(\hat{z}-z) \\ 0.02\,(\hat{\Psi}-\Psi) + 0 + 0 \end{pmatrix} \tag{25}$$

Here, $R(\Psi)$ denotes a planar rotation by Ψ. Observe that only hight control contains an integral component, while the yaw angle can well be controlled with pure proportional control.



(a) (b)

Figure 7: Model Parameter Estimation: (a) ground truth velocity (blue) and modelled velocity (red), which is estimated solely from the quadrocopter's attitude according to (26). The black line shows the accelerating force from (17) in x direction. (b) value of E_{c_1,c_2} (the darker, the smaller the error) for different c_1, c_2: a clear minimum is visible.

5.5. Model Parameter Estimation

To achieve accurate predictions, careful calibration of the model is required. We exemplary show the estimation of c_1 and c_2, which determine the effect of the quadrocopter's attitude on its horizontal velocity. The remaining parameters c_3 to c_8 are estimated analogously. For this we approximate the ground truth velocity from the motion capture system $\dot{x}(t)$ with the predicted model velocity

$$\hat{\dot{x}}_{t+\delta_t} := \hat{\dot{x}}_t + \delta_t\, \ddot{x}(\hat{\dot{x}}_t, \Phi_t, \Theta_t, \Psi_t), \tag{26}$$

which recursively predicts the quadrocopter's velocity based solely on its ground truth attitude, according to the model derived in (18). We then choose c_1 and c_2 as the minimizer of

$$E_{c_1,c_2}(c_1, c_2) := \sum_t \left(\dot{x}_t - \hat{\dot{x}}_t\right)^2, \tag{27}$$

using a generic, gradient-free minimization method. Figure 7 visualizes this error function and shows how well this simple, calibrated model approximates the flight dynamics of the quadrocopter.

6. Experiments and Results

We conducted a series of real-world experiments to analyze the properties of our approach. The experiments were conducted in different environments, i.e., both indoor in rooms of varying size and visual appearance as well as outdoor under the influence of sunlight and wind. A selection of these environments is depicted in Fig. 8. The pose and scale estimation accuracy was evaluated by attaching visual markers to the quadrocopter and tracking

[3] 18° for roll and pitch, 2 m/s for vertical and 90 °/s for yaw speed

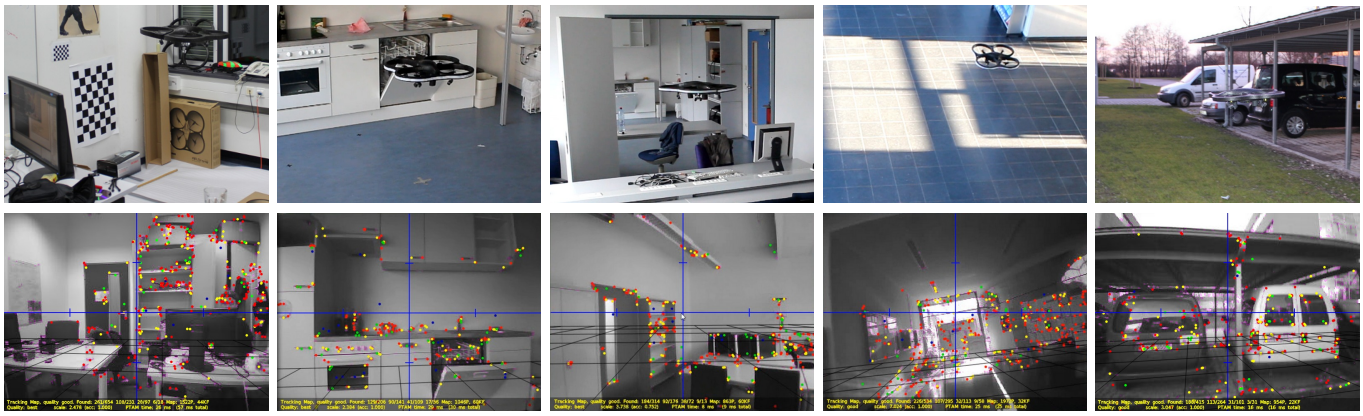


Figure 8: Testing Environments: The top row shows an image of the quadrocopter flying, the bottom row the corresponding image from the quadrocopter's frontal camera. This shows that our system can operate robustly in different, real-world environments.

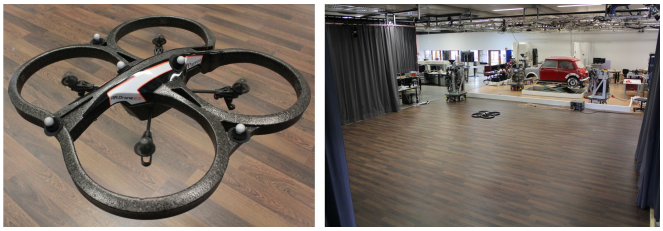


Figure 9: Motion Capture Setup: Left: AR.Drone with attached visual markers; Right: The motion capture volume. The externally tracked pose is solely used for evaluation purposes and at no point to control the quadrocopter.

it with an external Qualisys motion capture system, consisting of 11 cameras and covering a volume of roughly $4\,\mathrm{m} \times 4\,\mathrm{m} \times 2\,\mathrm{m}$ (Fig. 9). It allows to track the quadrocopters position and orientation at 50 Hz, with a global accuracy of less than a centimeter.

In this section, we first analyze the accuracy of the proposed ML scale estimator both in simulation and on real data, and compare it to an EKF-based approach in Sec. 6.1. We then give an evaluation of the complete system in terms of the pose estimation accuracy in Sec. 6.2 and the flight speed and stability of the quadrocopter in Sec. 6.3. In Sec. 6.4 we qualitatively demonstrate the robustness of our system to temporary loss of visual tracking. Finally, we give a brief discussion of the overall performance of the proposed system in Sec. 6.5.

6.1. Scale Estimation Accuracy

We evaluated the accuracy of the proposed scale ML estimator by comparing it to an EKF based approach as proposed in [20]: The state $\mathbf{x}$ consists of the current height z and vertical speed and $\dot{z}$, a bias term for the metric sensor b and the scale of the visual map λ. The observations are

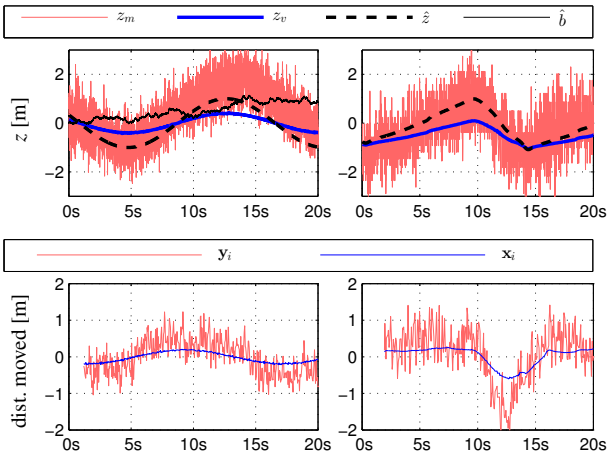


Figure 10: Data Examples: Left: synthetic data, Right: recorded test-flight data. The top plots show the raw visual and metric altitude measurements over time. The bottom plots show the corresponding sample pairs $(\mathbf{x}_i, \mathbf{y}_i)$.

given by the metric sensor readings as well as the visual height estimates, leading to the two observation functions

$$h_{\text{vision}}(\mathbf{x}) = \lambda z \qquad h_{\text{metric}}(\mathbf{x}) = z + b \tag{28}$$

To facilitate convergence, in our experiments we assured that the EKF is initialized with a scale that is at most 50% off.

We used (a) synthetic data and (b) real data from both an air pressure sensor and an ultrasound altimeter, which was obtained as follows:

(a) Synthetic Data: We assume that the quadrocopter flies a sinus-shaped trajectory, i.e. its true altitude in meters is given by

$$\hat{z}(t) := \sin(\alpha t), \tag{29}$$

where t is the time in seconds and α is chosen randomly between 0.2 and 1. Visual altitude measurements $z_v(t)$

are taken every 40 ms, while metric altitude measurements $z_m(t)$ are taken every 5 ms – both are subject to Gaussian white noise. We simulate a drift on the metric altitude measurements (corresponding to air pressure drift or uneven ground) as a Gaussian random walk $\hat{b}(t)$. The synthetic altitude and vision sensor data is hence generated by the following model:

$$z_v(t) \sim \mathcal{N}(\hat{\lambda}\hat{z}(t), \sigma_v^2) \tag{30}$$

$$z_m(t) \sim \mathcal{N}(\hat{z}(t) + \hat{b}(t), \sigma_m^2) \text{ with} \tag{31}$$

$$\hat{b}(t) \sim \mathcal{N}(\hat{b}(t - 1\,\text{ms}), \sigma_b^2) \tag{32}$$

(b) Real Data: We instructed the quadrocopter to repeatedly fly a distance of 2 m up and down, recording the visual pose estimates as well as readings from an ultrasound altimeter and an air pressure sensor. The ground truth scale $\hat{\lambda}$ was obtained by a 7 DoF (rigid body plus scale) alignment with the ground truth trajectory from the motion capture system. We repeated this experiment 10 times in different scenes with different depths. As we initialise the visual map such that the average keypoint depth is 1, the scale roughly corresponds to the inverse of the average scene depth on initialization – which in our experiments ranged from 3 m to 11 m.

Figure 10 shows a short extract of synthetic and real data, as well as the corresponding distance sample pairs which are computed as explained in Sec. 4.4.

The results of the experiments are visualized in Fig. 11: For all data sets, the proposed ML estimator converges quicker and to a more accurate value than the filtering approach. In particular for very large noise levels (second row), the EKF hardly converges at all (considering it is initialized with an error of at most 50%) and is slightly biased – while the proposed method still provides an accurate scale estimate after sufficient measurements have been integrated. It is to mention that for this extreme case, the noise standard deviations $\sigma_m = 6$ m and $\sigma_v = 0.3$ is extremely large compared to the small height interval of only 2 m (0.5 in vision units) used by the quadrocopter.

Failure Modes: In our experiments we found the EKF to be sensitive to certain parameters, such as the prediction uncertainty on z and $\dot{z}$ (corresponding to the unmodeled acceleration, i.e., the quadrocopter's flight pattern), and the prediction uncertainty on λ. In general there seems to be a trade-off between very slow convergence on the one hand, and significantly biased results, oscillating behaviour or even divergence on the other.

The proposed ML estimator fails if $s_{xy} \approx 0$, which may happen in the beginning if the distance travelled within the measured intervals is always significantly smaller than

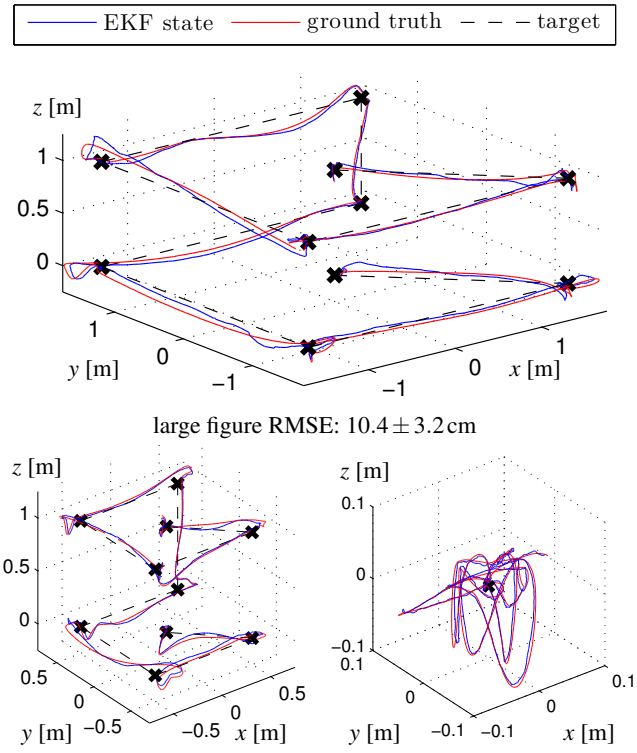


Figure 12: Pose Estimation Accuracy: Estimated trajectory (blue) and ground truth trajectory (red) for three flights: a large figure (3 m × 3 m × 1 m), a small figure (1 m × 1 m × 1 m), and holding a position.

the measurement noise – in fact, the magnitude of s_{xy} is a good indicator of the estimator's accuracy. The resulting unstable behaviour can be observed during the first 50 s of the last example in Fig. 11 – this simply means that no reasonable scale estimate is feasible from the collected data. This can be resolved by increasing the windows size k as introduced in 4.4, or by introducing a prior λ_0 as additional sample pair $(\mathbf{x}_0, \mathbf{y}_0) := (w\lambda_0, w)$, where w is the prior's weight.

For real-world data using an ultrasound altimeter, the average scale estimation error of the ML estimator is 5% after 3 s, and 1% after 20 s Using an air pressure sensor, it is 20% after 10 s, and 6% after 30 s, which is significantly better to what can be achieved from inertial measurements alone. On the other hand, it restricts the flight pattern to include sufficient vertical motion in the first seconds after map initialization, which in practice however can easily be enforced. Further – when using the air pressure sensor – a decent initial guess or prior is required as the estimated scale can be very inaccurate during the first seconds.

6.2. *Pose Estimation Accuracy*

We analyzed the accuracy of the quadrocopter's pose estimate after all sensor information up to that point in

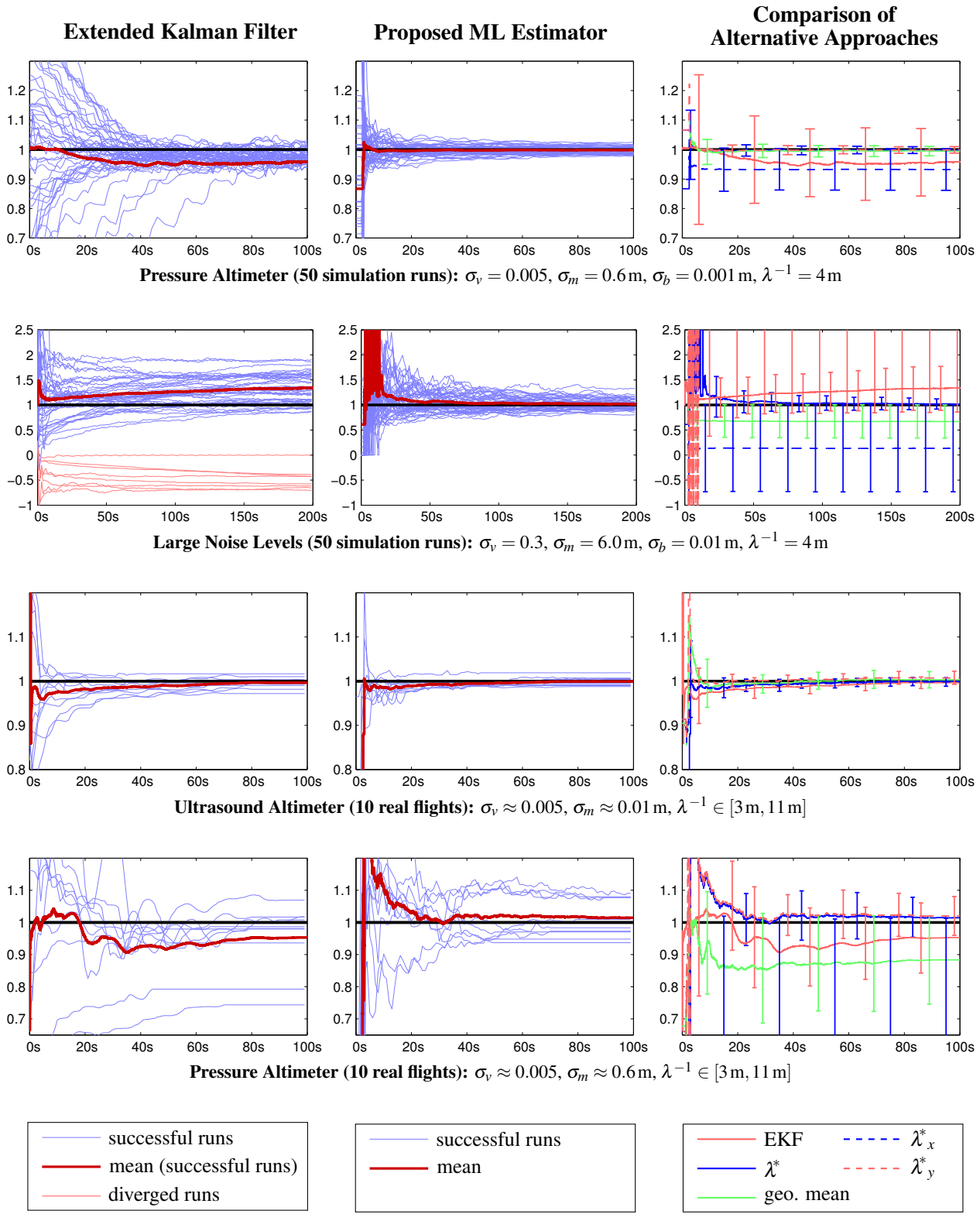


Figure 11: Scale Estimation Comparison: The plots show the result of the different scale estimation methods over time. Each row corresponds to a different dataset. The left plots show the EKF scale estimates for the individual runs, as well as their average in red. For very large noise levels, the EKF scale estimate occasionally diverges – these runs are ignored and visualized in red. The middle plots show the proposed ML estimator for the individual runs. The right plots show the mean and standard deviation for the other estimators defined in Chapter 4.2. Note the different scales on the plots.

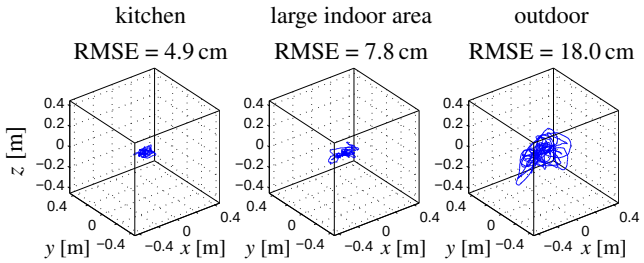


Figure 13: Flight Stability: Path taken and RMSE of the quadrocopter when instructed to hold a target position for 60 s, in three of the environments depicted in Fig. 8. It can be seen that the quadrocopter can hold a position very accurately, even when perturbed by wind (right).

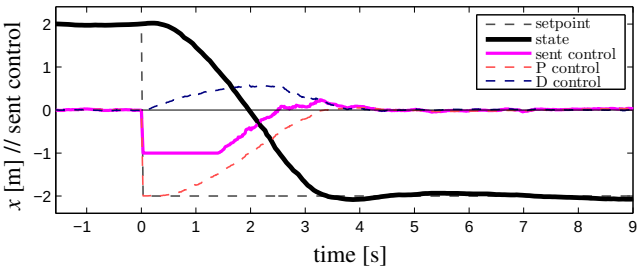


Figure 14: Control Behaviour: The plot shows the behaviour of the quadrocopter when flying a large distance. As can be seen, the quadrocopter accelerates with maximum roll for the first second and decelerates before converging on the set-point. The dashed red and blue lines show the proportional and differential control components respectively.

Table 1: Measured flight and convergence speed for position control

distance	peak flight speed	convergence time
$(1,0,0)^T$ m	0.9 ± 0.1 m/s	2.4 ± 0.4 s
$(3,0,0)^T$ m	2.0 ± 0.1 m/s	3.1 ± 0.8 s
$(0,0,1)^T$ m	0.5 ± 0.2 m/s	3.7 ± 0.1 s
$(1,1,1)^T$ m	1.1 ± 0.2 m/s	3.9 ± 0.5 s

time – including vision – has been integrated, which is therefore only available with a delay of roughly 250 ms. At this point in time, the visual pose estimates dominate the EKF state due to their comparatively high accuracy.

We measured the pose estimation accuracy for three different scenarios: flying a large figure utilizing the full motion capture volume, flying a small figure, and holding a given position, i.e. staying within a very small volume. We then performed a 6 DoF alignment between the quadrocopter's estimated trajectory and the ground truth trajectory. Figure 12 shows an example flight for each of the three scenarios, as well as the root mean square (RMSE) error over five independent flights per scenario.

It can be observed that the pose estimation error relates linearly to the size of the covered volume: while locally the quadrocopter's movement is estimated very accurately, over large distances the estimation error becomes larger. Note that – as PTAM always tracks against a global map and not frame-to-frame – the pose estimation error does not increase with the path length, but depends on the size of the covered volume.

6.3. Positioning Accuracy and Flight Speed

We evaluated the performance of the complete system in terms of position control. In particular, this reflects the effectiveness of the employed delay compensation (Sec. 5.3): without these measures, delays in the control loop quickly cause oscillations and unstable flight performance, requiring low control gains (i.e., cause slow flight). In particular, we measured the average time taken to approach a given goal location and the average positioning error while holding this position. Considering the large delay in our system, the pose stability of the quadrocopter heavily depends on an accurate prediction from the EKF: the more accurate the pose estimates and in particular the velocity estimates are, the higher the controller gains can be set without leading to oscillations.

To determine the stability, we instructed the quadrocopter to hold a target position over 60 s in different environments and measured the root mean square error of the estimated trajectory. Figure 13 shows the result for three different environments: the measured RMSE lies between 4.9 cm (indoor) and 18.0 cm (outdoor).

To evaluate the flight speed, we repeatedly let the quadrocopter fly a given distance and measured the convergence time, that is the time required until the Euclidean distance to the target position falls and stays below 10 cm. An example of flying a long distance in x-direction is shown in Fig. 14: the plot clearly shows that the quadrocopter accelerates initially with maximum roll, and actively decelerates before reaching the target location at $t = 3.5$ s. Figure 15 shows position and speed of the quadrocopter over time for the large figure displayed in Fig. 12, note how quickly and accurately the quadrocopter flies from set-point to set-point. Table 1 shows the average time required to move a given distance: reaching a target location at a distance of 3 m for example takes 3.1 s on average, with the quadrocopter accelerating up to a speed of 2 m/s.

6.4. Robustness to Temporary Loss of Visual Tracking

The system as a whole is robust to temporary loss of visual tracking, e.g., due to occlusions or large rotations,

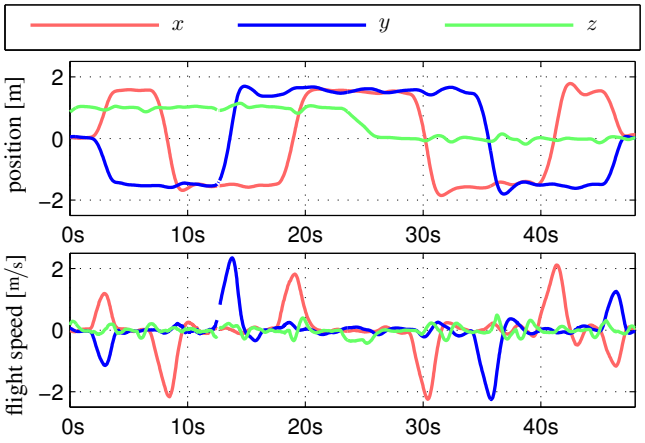


Figure 15: Example Flight: This plot shows the ground truth velocity and position of the large figure flight shown in Fig. 12, illustrating the typical behaviour of the quadrocopter when holding and approaching way-points. Note how the quadrocopter accelerates up to a horizontal speed of 2 m/s when instructed to fly a distance of only 3 m.

as it continues to navigate based only on odometry and IMU measurements. As soon as visual tracking recovers, the EKF state is updated with the absolute pose estimate, eliminating accumulated estimation error. Figure 16 shows an extract from a flight where visual tracking is lost temporarily, due to the quadrocopter being pushed and rotated away from its target position.

6.5. Discussion

We demonstrated our system repeatedly and with great success at various events and to external visitors, and generally observed good and very reliable performance.

One weakness is the heavy reliance on wireless LAN communication, which causes problems in the presence of many wireless-capable devices like smartphones. This can be resolved by running all computations onboard, which would require more sophisticated hardware.

Second, our approach requires a suitable visual environment: As it relies heavily on the frontal camera, a sufficient amount of structure / texture at an adequate distance (in our experience 2-15 m) is required in its field of view. For ultrasound-based scale estimation, a flat ground surface is assumed – sudden jumps (e.g., flying over a table) are detected as outlier and filtered out. Depending on the flight-pattern, a sloped ground surface can bias the scale estimation result. This however is rarely the case indoors, while outdoors the pressure altimeter poses a suitable alternative.

Third, as for all monocular SLAM methods, PTAM cannot handle large rotation (yaw) without sufficient simultaneous translation, such that newly seen parts of the environment can be triangulated.

7. Conclusion

We presented a visual navigation system for a low-cost quadrocopter with off-board processing. Our approach enables the quadrocopter to visually navigate in unstructured, GPS-denied environments and does not require artificial landmarks nor prior knowledge.

The contribution of this paper is two-fold: first, we presented a robust solution for visual navigation of a low-cost quadrocopter with off-board computation. Second, we derived a maximum-likelihood estimator in closed form to recover the absolute scale of the visual map, which is an efficient and more accurate alternative to existing filtering-based methods.

We extensively tested and evaluated our system, among other with respect to its pose estimation accuracy (10 cm error over a $3 \times 3 \times 1$ m volume) and control accuracy (4.9 cm RMSE indoor and 18.0 cm outdoor). We experimentally showed that the derived scale estimation method accurately estimates the scale of the visual map and can be used with different metric sensors, and how it compares to a state-of-the-art filtering-based approach – both in simulation and on real data.

We made the complete implementation available as an open-source ROS package *tum_ardrone*, with the aim to facilitate the reproduction of our results and to stimulate future research projects on such platforms.

References

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[2] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," in *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[3] R. Ritz, M. Mueller, and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching," in *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[4] A. Kushleyev, D. Mellinger, and V. Kumar, "Towards a swarm of agile micro quadrotors," in *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[5] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009.

[6] M. Achtelik, S. Lynen, S. Weiss, L. Kneip, M. Chli, and R. Siegwart, "Visual-inertial SLAM for a small helicopter in large outdoor environments," in *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[7] F. Fraundorfer, L. Heng, D. Honegger, G. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor MAV," in *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

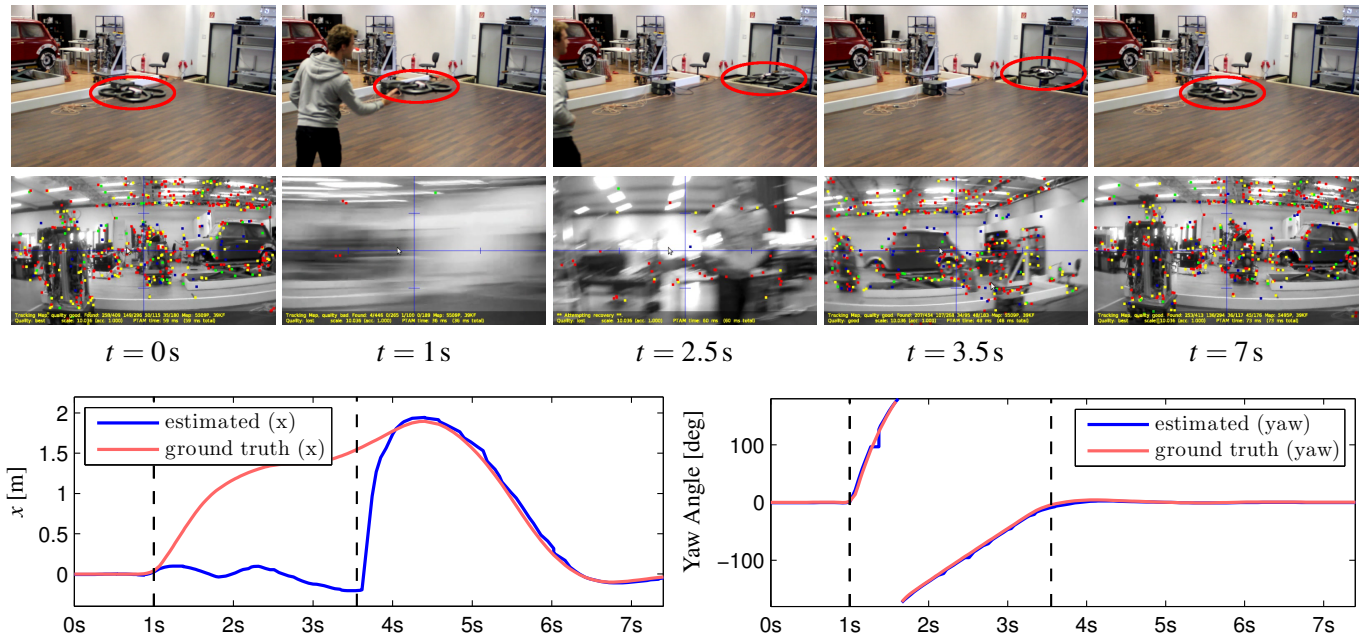


Figure 16: Robustness to Visual Tracking Loss: The quadrocopter is instructed to hold a flying position. At $t = 1\,\mathrm{s}$ it is pushed and rotated away, such that visual tracking gets lost. Using IMU measurements, the quadrocopter tries to fly back to the goal location, in particular it corrects its yaw orientation. At 3.5 s visual tracking recovers and the quadrocopter flies back to the real target position. Note how the estimated state quickly converges to the true value, as soon as visual tracking recovers.

[8] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. IEEE Intl. Symposium of Robotics Research (ISRR)*, 2011.

[9] E. Bylow, J. Sturm, C. Kerl, F. Kahl, D. Cremers, "Real-time camera tracking and 3D reconstruction using signed distance functions," in *Proc. of Robotics: Science and Systems (RSS)*, 2013.

[10] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[11] ——, "Accurate figure flying with a quadrocopter using onboard visual and inertial sensing," in *Proc. of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[12] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[13] D. Eberli, D. Scaramuzza, S. Weiss, and R. Siegwart, "Vision based position control for MAVs using one single circular landmark," *Journal of Intelligent and Robotic Systems*, vol. 61, pp. 495 – 512, 2011.

[14] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-drone as a platform for robotic research and education," in *Proc. Research and Education in Robotics: EUROBOT 2011*, 2011.

[15] "Ascending technologies," 2013. [Online]: http://www.asctec.de/

[16] M. Achtelik, A. Bachrach, R. He, S. Prentice, and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments," in *Proc. SPIE Unmanned Systems Technology XI*, 2009.

[17] K. Schmid, F. Ruess, M. Suppa, and D. Burschka, "State estimation for highly dynamic flying systems using key frame odometry with varying time delays," in *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[18] V. Grabe, H. Bulthoff, and P. Giordano, "Robust optical-flow based self-motion estimation for a quadrotor UAV," in *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[19] P. Bristeau, F. Callou, D. Vissière, N. Petit, *et al.*, "The navigation and control technology inside the AR. drone micro UAV," in *World Congress*, 2012.

[20] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[21] S. Yang, S. A. Scherer, A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle," *Journal of Intelligent and Robotic Systems*, vol. 69, pp. 499 – 515, 2013.

[22] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2012.

[23] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[24] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-drone as a platform for robotic research and education," in *Proc. Communications in Computer and Information Science (CCIS)*, 2011.

[25] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots," in *Proc. IEEE Intl. Symposium on Robot and Human Interactive Communication*, 2011.

[26] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[27] J. Engel, "Autonomous Camera-Based Navigation of a Quadrocopter," Master's thesis, Technical University Munich, 2011.